Midterm Report
Content-Aware Seam Carving

Chenguang Ji
CS6475 Fall 2020
cji62@gatech.edu

***Abstract*—This assignment aims to understand and replicate the results of two papers on seam carving for content-aware image resizing, which focuses on backward energy method and forward energy method respectively. In this report, I will introduce the basic algorithm to realise content-aware image resizing and compare my results to the research paper outputs qualitatively, and quantitatively using the metrics that I develop. At the end, I will discuss the difficulties in replicating the results.**

***Index Terms*—seam carving, content-aware, backward, forward, energy**

I. Discussion of Algorithm

The general algorithm is similar for both backward energy method and forward energy method. To simplify the discussion, backward energy method is used as an example, and then subtle differences in the forward energy method will be explained. Parameters including border condition, input format, choice of energy function, etc. can have huge effect on the quality of the output, and they will be discussed in Section IV and the video presentation.

Assume that we want to reduce the width of an image I by 1. First we calculate the energy map of I, which represents the energy of each pixel in I. And then we use dynamic programming to construct a cumulative path map, which represents the cumulative minimum energy for all possible connected seams of each pixel. The next step is to backtrack from the minimum value of the last row to find the minimum energy seam. The seam is removed from the image to get the reduced image (I^{-1}), and the seam information will be stored. To reduce the width by k, we can simply iterate the above process k times. For iteration i, the input image is $I^{-i}$and the output will be $I^{-(i+1)}$.

To enlarge an image I by k, first we need to identify the first k minimum energy seams for removal, and then insert the seams. Finding the seams can be achieved by the reducing algorithm above. What we need is the seam information during each reducing iteration. However, the seam information we get from iteration i only represents the seam in I^{-i}, and does not show its position on the original image. To solve this, each seam will be compared with all the seams removed prior to it and adjusted accordingly. The seams can then be inserted to the original image, or we can compare the original input with the seams, and construct the enlarged image pixel by pixel.

In the case of forward energy method, instead of finding the seam with the least energy, the seam whose removal will insert the minimum energy to the image is identified. A so-called cost map, is constructed using values of the pixel intensities of the image, so there is no need to calculate an energy map.

Algorithm 1: Reduce and Enlarge an Image by k

function SeamRemoval (img);
Input : Original image I
Output: Image with reduced width I^{-k}, **seams**
for *i in range(0, k)* **do**
  Calculate energy map e(I^{-i}) for I^{-i}
  Calculate cumulative minimum energy M
  Backtrack the minimum energy seam
  Store seam information in **seams**, remove the seam from I^{-i} to get $I^{-(i+1)}$
  Input for next iteration = $I^{-(i+1)}$
end
return I^{-k}, **seams**
function SeamInsertion (img);
Input : Original image I
Output: Image with enlarged width I^{+k}
Run SeamRemoval, get **seams**
Sort **seams**
for *i in range(0, k)* **do**
  Duplicate seam i from I^{+i} to get $I^{+(i+1)}$
  Input for next iteration = $I^{+(i+1)}$
end
return I^{+k}
Or
for *I[i,j] in I* **do**
  if I[i,j] not in **seams**
  I^{+k}[y,x] = I[i,j]
  else
  I^{+k}[y,x] = I[i,j]
  I^{+k}[y,x+1] = 0.5*(I[i,j] + I[i,j+1])
end
return I^{+k}

II. Comparison Metrics

I use two comparison metrics, one is average energy ratio (**R**). It is defined as the ratio of the average pixel energy of two images, in the range of 0-1, while 0 represents the largest differences and 1 represents the smallest differences. Because both forward and backward energy method aim to introduce

the minimum energy change to the image. By comparing the average energy of two outputs, we can know if the two process work equally well in preserving important contents in terms of energy. Similar metrics is also used in **Ref. 1**.

The second metric is histogram correlation (**C**). It is defined as the correlation of histograms of two images,[3] in the range of 0-1, while 0 represents the lowest similarity and 1 represents the highest similarity. Histogram gives us the intensity distribution of an image.[4] Identical histograms are no sufficient condition for identical images, however in our case, two different outputs are derived from the same input by removing/adding seams and without large scale swapping of pixels. So it is a good way to tell if pixels with similar intensity have been removed/added during the process.

III. Replicated Images and Comparison of Results

A. Fig. 5 Waterfall (2007) Backward Energy Seam Removal

My waterfall output matches the reference output, with small visible differences. **R** is 0.998 and **C** is 0.995, which means when compared to the reference output, important contents are preserved and similar pixels have been removed.

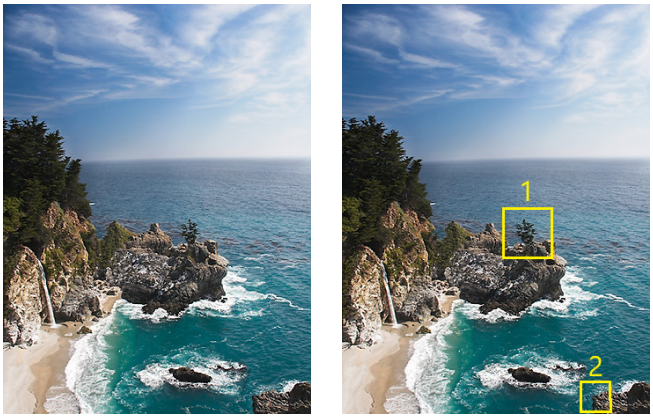

Fig. 1. Left : student output. Right: reference output. Differences are marked with yellow boxes.

B. Fig. 8 Dolphin (2007) Backward Energy Seam Insertion

Generally, my three outputs matches well with the reference outputs. From the seam output we can see the major differences are in region 1. In my seam output, there is a seam to the left of the fin, and there are more seams to the right, which are closely packed. The extra seam causes a small artifact in region 1 of both 50% and 2-step 50% dolphin outputs. However, the closely packed seams to the right of the fin gives smoother result in region 2, while in the reference dolphin output, there is a visible dent. **R** of the three output pairs are 0.948, 0.999 and 0.998, respectively. And **C** are 0.999, 0.998 and 0.997, respectively. Which confirms that my outputs are very similar to reference outputs, both in energy preservation and how pixels are added. It is worth noting that, for the seam output, the average energy ratio is not as satisfactory as the histogram correlation, which is because differences in the seam distribution will change the energy of the image, while it has almost no effect on the per-pixel based histogram if similar pixels are added anyway.

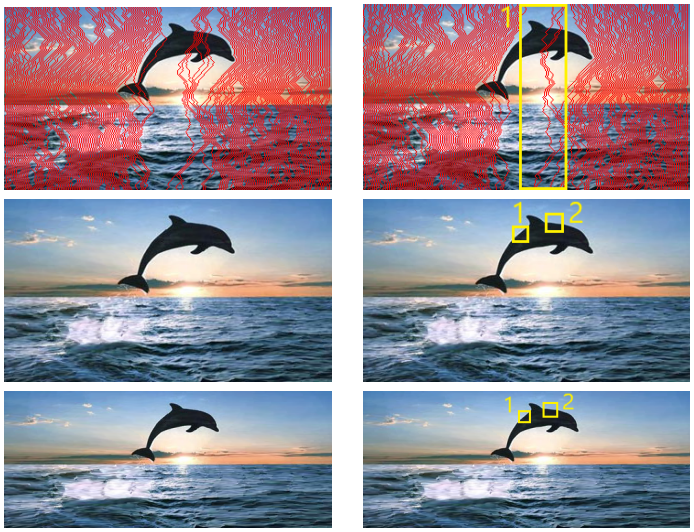

Fig. 2. Left: student outputs. Right: reference outputs. From top to bottom: 50% enlarged dolphin image with red seams added; 50% enlarged dolphin image and 2-step 50% enlarged dolphin image. Differences are marked with yellow boxes.

C. Fig. 8 Bench (2008) Seam Removal by Two Methods

For the backward energy method, one major difference is there are less seams in region 1 of my seam output, which intersect the bench. The visual result it, waterdrop features are almost removed in the reference bench output (as shown in region 2 of the reference output), while my output preserves such features well. Because of the differences in seam distribution, **R** and **C** of backward seam outputs are 0.891 and 0.952. However, **R** and **C** of backward bench output are 0.993 and 0.996, indicating that slight differences in seams does not effect the overall feature preservation.

For the forward energy method, the seam density is different in region 1 and 2. The bench in my bench output is shorter than tha of the reference bench output. According to **Ref. 5**, the feature in region 3 of the reference bench output is missing when compared to the corresponding seam output. **R** and **C** of forward seam outputs are 0.856 and 0.932. **R** and **C** of forward bench output are 0.905 and 0.981. These metrics are the worst among all my outputs, which indicates there are more discrepancies in how the images are reduced, both in energy preservation and pixels removal.

More importantly, the backward energy method generates outputs with significant artifacts while the forward method yields decent result. It shows the advantages of the forward energy method when the energy change is dominated by the energy inserted when removing a seam.

D. Fig. 9 Car (2008) Seam Insertion by Two Methods

For the backward energy method, both outputs are severely distorted. Region 1 and 2 are examples of such distortion. For the forward energy method, there are still some abnormally

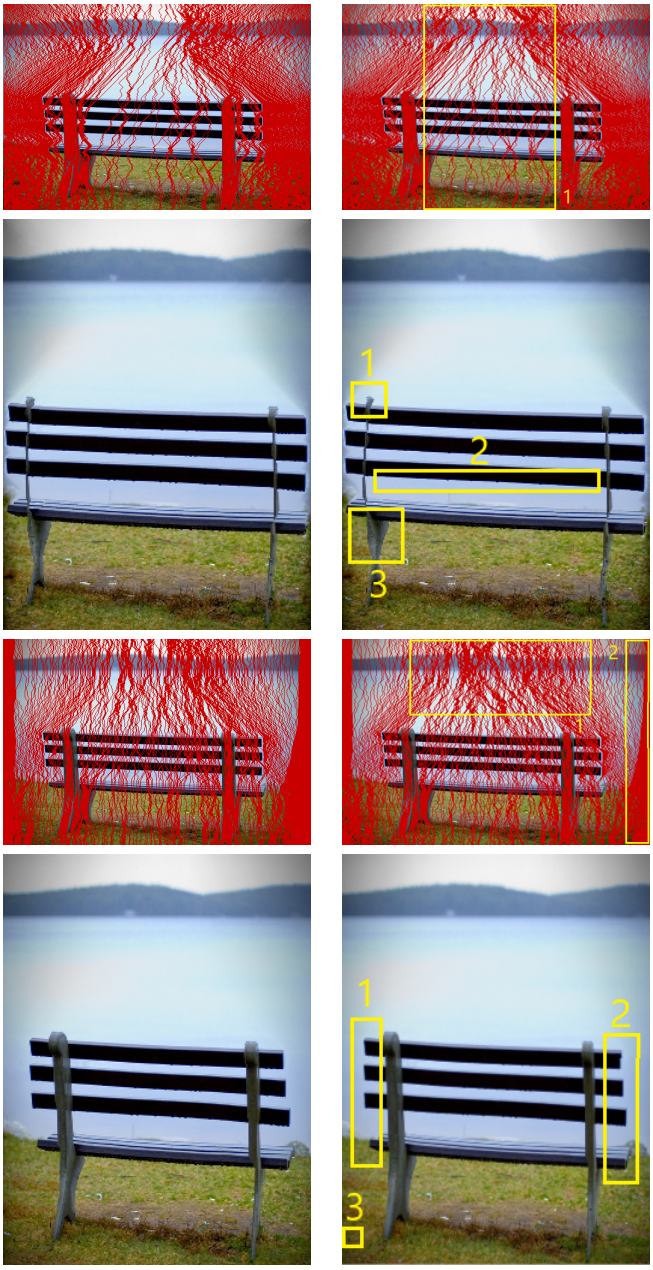

Fig. 3. Left: student outputs. Right: reference outputs. From top to bottom: bench image with red seams to be removed by backward energy; 50% reduced bench image by backward energy; bench image with red seams to be removed by forward energy and 50% reduced bench image by forward energy. Differences are marked with yellow boxes.

stretched regions in my output as well as the reference output, but they are much better compared to the backward method. Region 1 is stretched in the reference output but looks better in my output, while in region 2, the reference output is smoother. **R** and **C** of backward outputs are 0.999 and 0.999, respectively. **R** and **C** of forward outputs are 0.994 and 0.999. These metrics are the best among my results, which indicates the images are fairly similar, and they do look so.

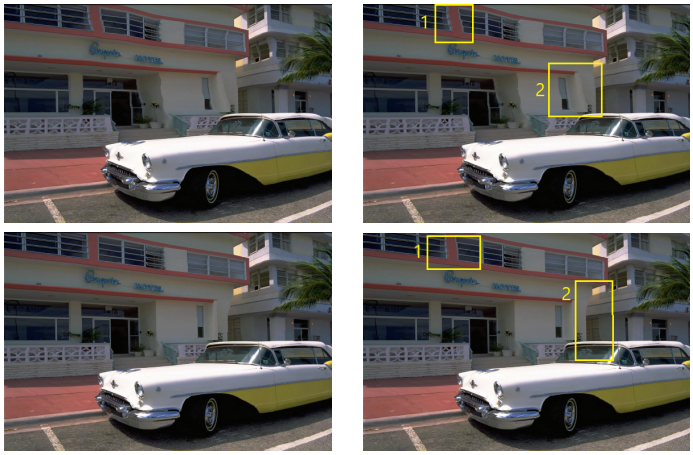

Fig. 4. Left: student outputs. Right: reference outputs. From top to bottom: 50% enlarged car image by backward energy and 50% enlarged car image by forward energy. Differences are marked with yellow boxes.

IV. Ambiguities and Issues

In both papers,[1,2] the inputs are color images. However, when we calculate the energy/cost of a pixel, we need to get an accordant result for three color channels. A natural solution is to use the average of three channels. To do so, we can either convert the original input to gray scale first, or we can calculate the average among three channels afterwards. These two approaches can produce different outputs, and in extreme scenarios, the differences can be huge.

In the 2007 paper, although the authors specify several energy function to measure the importance of image pixels,[1] it is unclear how they calculate the energy value: what the kernel size is, do they use any border padding, do they use L1 or L2-norm of the gradient?

In the 2008 paper, The cost and accumulative cost matrix are defined in **Section 5.1**,[2] but the image edge cases are not clearly defined, making it impossible to calculate without extra border padding or special edge handling.

To solve these, I need to make my own assumptions and definitions. Since it is hard to find a universal pipeline that works perfectly for every input, I tried to replicate the reference outputs by a trial & error method. I used different set of parameters, and compared my outputs with the reference. Importantly, I made theoretical predictions before I try each parameter, which narrowed my choices and made this process more efficient.

Examples of how different parameters can change the outputs will be covered in the video presentation.

V. Video Presentation

Here you can see my presentation of this report and some of the work I did in a visual format: https://www.dropbox.com/s/67yij7al5g5d7jy/Video%20Presentation.mp4?dl=0

References

[1] S. Avidan, A. Shamir "Seam Carving for Content-Aware Image Resizing", 2007.

[2] M. Rubinstein, S. Avidan, A. Shamir “Improved Seam Carving for Video Retargeting”, 2008.
[3] OpenCV, histogram comparison, https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html, Accessed: Oct. 15, 2020
[4] OpenCV, histograms, https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html, Accessed: Oct. 15, 2020
[5] Piazza post @861, https://piazza.com/class/kdkh6c8vx7z5tf?cid=816, Accessed: Oct. 15, 2020
[6] Piazza post @786, https://piazza.com/class/kdkh6c8vx7z5tf?cid=786, Accessed: Oct. 15, 2020
[7] Piazza post @740, https://piazza.com/class/kdkh6c8vx7z5tf?cid=740, Accessed: Oct. 15, 2020
[8] Numpy, numpy.argmin, https://numpy.org/doc/stable/reference/generated/numpy.argmin.html, Accessed: Oct. 15, 2020
[9] Numpy, numpy.argsort, https://numpy.org/doc/stable/reference/generated/numpy.argsort.html, Accessed: Oct. 15, 2020
[10] OpenCV, image gradients, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_gradients/py_gradients.html, Accessed: Oct. 15, 2020